ERDC MSRC Major Shared Resource Center

ERDC MSRC PET Technical Report No. 01-15

CFDTool: A Web-based Training Tool for CFD

by

Roy P. Koomullil
Bharat K. Soni

4 May 2001

Work funded by the Department of Defense
High Performance Computing Modernization Program
U.S. Army Engineer Research and Development Center
Major Shared Resource Center through

Programming Environment and Training

Supported by Contract Number: DAHC94-96-C0002
Computer Sciences Corporation

CFDTool: A Web-based Training Tool for CFD

Roy P. Koomullil[*] and Bharat K. Soni[§]

1 Introduction

The rapid development of supercomputers, high performance workstations, numerical algorithms, and high-speed data networks has resulted in dramatically increased computational power and efficiency. As a result, Computational Fluid Dynamics (CFD) has emerged as an essential analysis tool, and has fundamentally changed the way in which the underlying principles of science and engineering are applied to research, design and development problems. CFD techniques have been used extensively in analyzing fluid mechanics, heat and mass transfer, electromagnetics, hydrodynamics, atmospheric sciences, solid mechanics, water quality and effluent transport problems, as well as many other problems.

In the CFD process, the partial differential equations (PDEs), which govern the problem of interest, are solved using numerical methods on a high-speed computer or network of computers. Many methods have been used for approximating the continuous PDEs by discrete numerical approximations. Among these methods are finite-difference, finite-volume, finite-element, integral and spectral methods. The CFD process can be described by the following steps:

Step I: Pre-processing
- Problem definition
- Selection of governing equations
- Domain decomposition
- Geometry discretization
- Numerical grid generation
- Boundary condition specification

Step II: Processing
- Solve a set of discretized PDEs using numerical algorithms

Step III: Post-processing
- Graphical visualization and interpretation of simulated field characteristics

The different phases of the simulation process - such as grid generation, geometry definition, flow simulation, and visualization - depend, in part, on information obtained in other phases. This information-passing results in a time-consuming trial and error procedure that requires an experienced user. User-friendly graphical user interfaces (GUIs) have the potential for seamlessly coupling the various phases, thus lessening the

[*] Research Engineer I, Center for Computational Systems, Engineering Research Center, Mississippi State University, MS 39762, email: roy@erc.msstate.edu
[§] Director, Center for Computational Systems, Engineering Research Center, Mississippi State University, MS 39762, email: bsoni@erc.msstate.edu

need for highly specialized users and streamlining the procedure. These GUIs can be also used for teaching newcomers the overall process of CFD.

One of the missions of the Major Shared Resource Center (MSRC) Programming Environment and Training (PET) program is to educate students and Department of Defense (DoD) High Performance Computing (HPC) practitioners in solving large-scale simulation problems. Several steps have been taken towards this goal. Under Army Research Laboratory (ARL) and Aeronautical Systems Center (ASC) sponsorship, a GUI-based training course in grid generation is being developed at Mississippi State University (MSU). Also a flow solver demonstration tool was created using the WEB-based tool, jvwt (java virtual wind tunnel), which was developed at Massachusetts Institute of Technology (MIT) and enhanced at MSU to demonstrate the CFD simulation process to undergraduate students at Jackson State University (JSU), Mississippi. However, a more thorough GUI driven training and demonstration tool is needed to teach and demonstrate basic concepts of CFD and HPC to students, young engineers and non-CFD DoD practitioners. This development, called CFDTool, provides a basis for future development of WEB-based training tools for CFD practitioners.

The CFDTool uses the Qt GUI library (http://www.trolltech.com/) and OpenGL to provide interactive visualization and control to various selections (Boundary conditions, geometry, grid, flow variables to visualize, numerical scheme to exercise, dissipation order to apply, angle of attack, contour vs. line plots, etc.). The software contains various numerical schemes, boundary conditions and macros for geometry and grid information.

The CFDTool consists of many modules that are explained in detail in Section 2. The methodology used for the grid generation and basic algorithms used to solve the governing equations are described in Section 3 and 4, respectively, and are followed by references in Section 5.

2 Modules Description

The CFDTool is a web-based program that allows the user to go through the various steps involved in CFD analysis. These steps involve: geometry definition, grid generation, boundary condition set-up, governing equations solution, and visualization. These modules are combined together using a main control unit as shown in Figure 1.

From the geometry module, the user can load a geometry definition file (Figure 2 and 3) as a set of points. The set of points can be either in the PLOT3D format or co-ordinates of points as *(x,y,z)* pairs. The geometry file can be either the salient points defining the geometry or can be a discretization of the geometry with appropriate point distribution for the simulation. In either case, the user can redistribute points (Figure 4) in the geometry after loading the geometry file. This redistribution is achieved by interactively selecting a segment of the geometry and specifying the number of points required and the type of distribution desired. Options are provided to distribute points using different distribution functions such as exponential and hyperbolic functions. Point clustering can be specified on either ends of the curve, both ends of the curve, or anywhere in between. Snapshots of the modules related to geometry are given in Figures 2-4.

The next step in the flow simulation process is setting up the boundary condition, and is achieved by utilizing the boundary condition module (Figure 5). In this module, users can select different boundaries interactively and set boundary conditions for that segment. All the boundary conditions are color-coded and the color of the boundary curve changes when the user sets a new boundary condition. The default boundary condition is set as a slip wall.

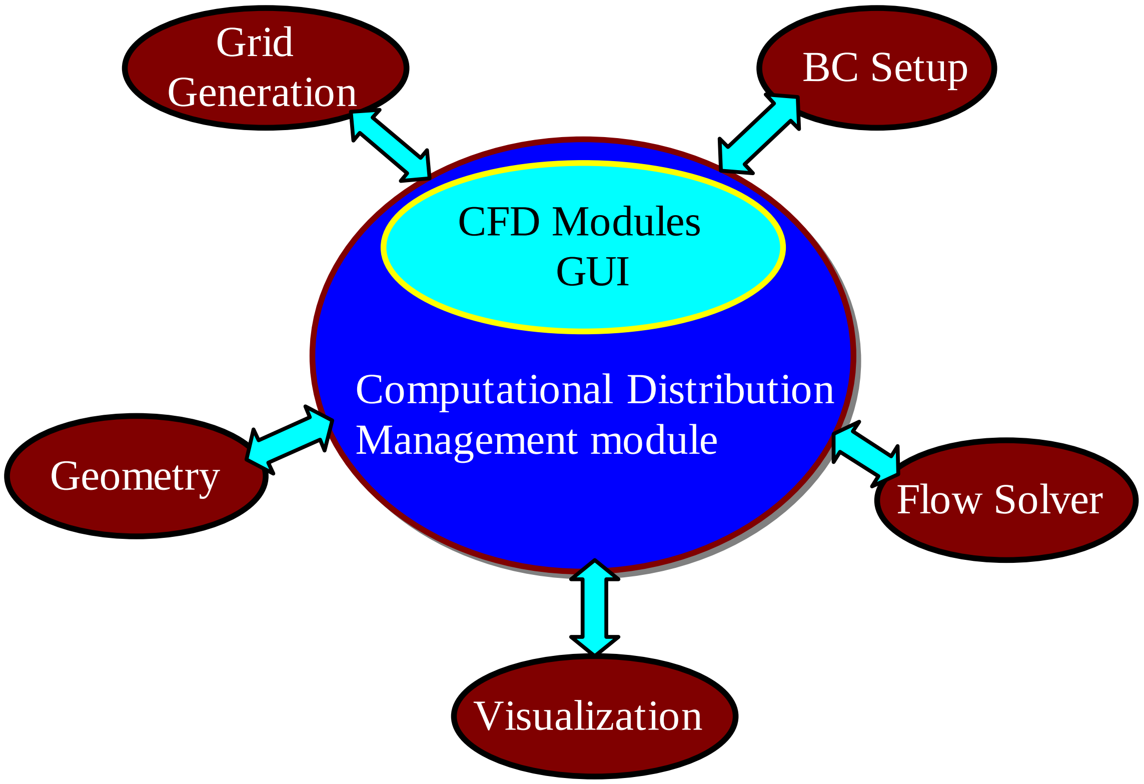


Figure 1. Different Modules in CFDTool

The next step in the process is the grid generation. Two different types of grids can be generated using CFDTool: unstructured and hybrid grids. The different parameters of the grid generators, such as the point distribution, can be changed using the grid module of the GUI. The input to the unstructured grid generator is a stretching parameter (Figure 6), which controls the shape of the generated triangles. For the hybrid grid, the input is the function that defines the normal growth of the point distribution. The approach for the hybrid grid generation is explained in Section 3.

In the flow solver module (Figure 7), the user can select different schemes, governing equations, accuracy, number of time-steps, etc. and set the freestream conditions. In the current version of the program the user has to wait until the end of the time iteration to visualize the results. We are looking into different options to visualize intermediate results. In the visualization module (Figure 8), the user has the option to select any of the conserved variables for visualization purposes. Currently, the results are presented as shaded contours. Pictures showing the GUI for grid, flow simulation, and visualization modules are shown below:

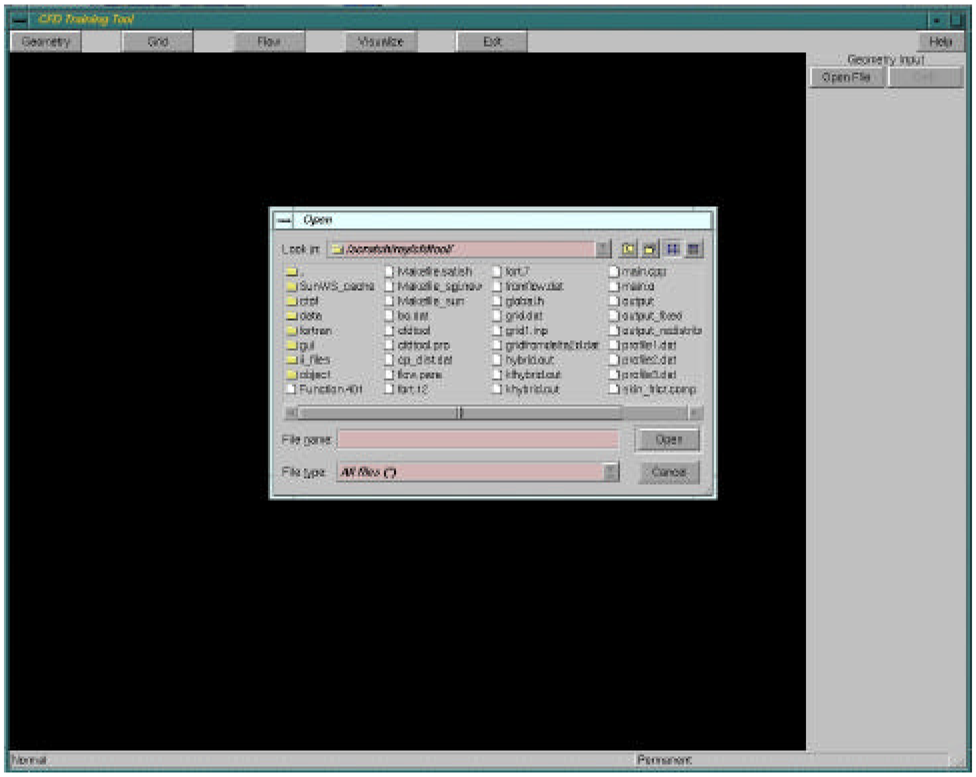


Figure 2. GUI for Reading Geometry

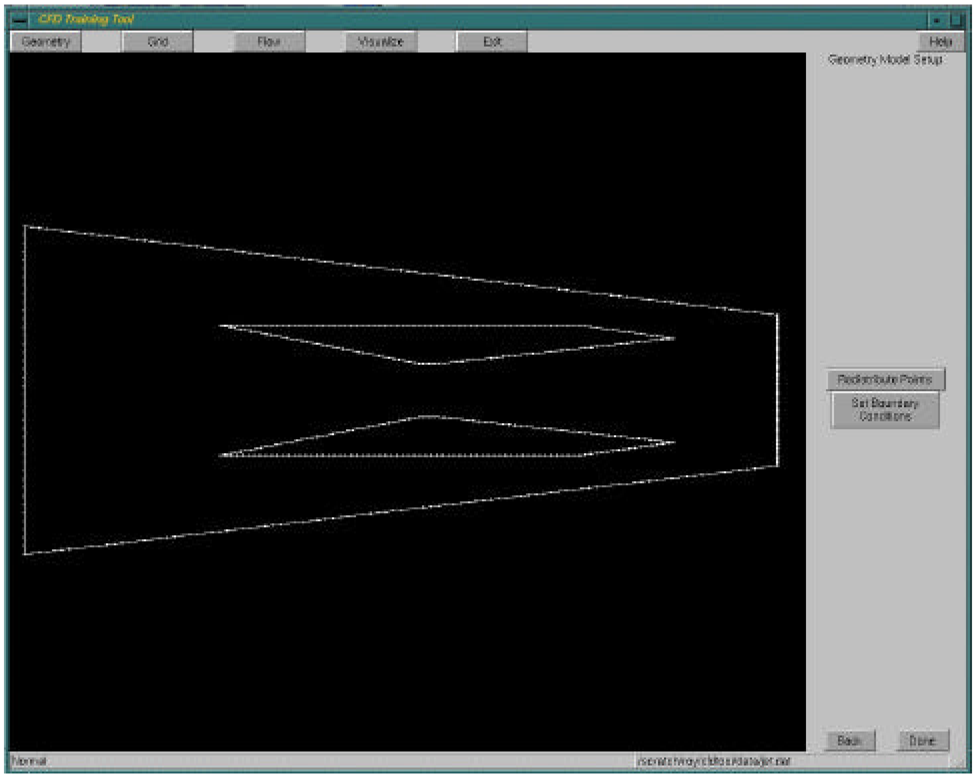
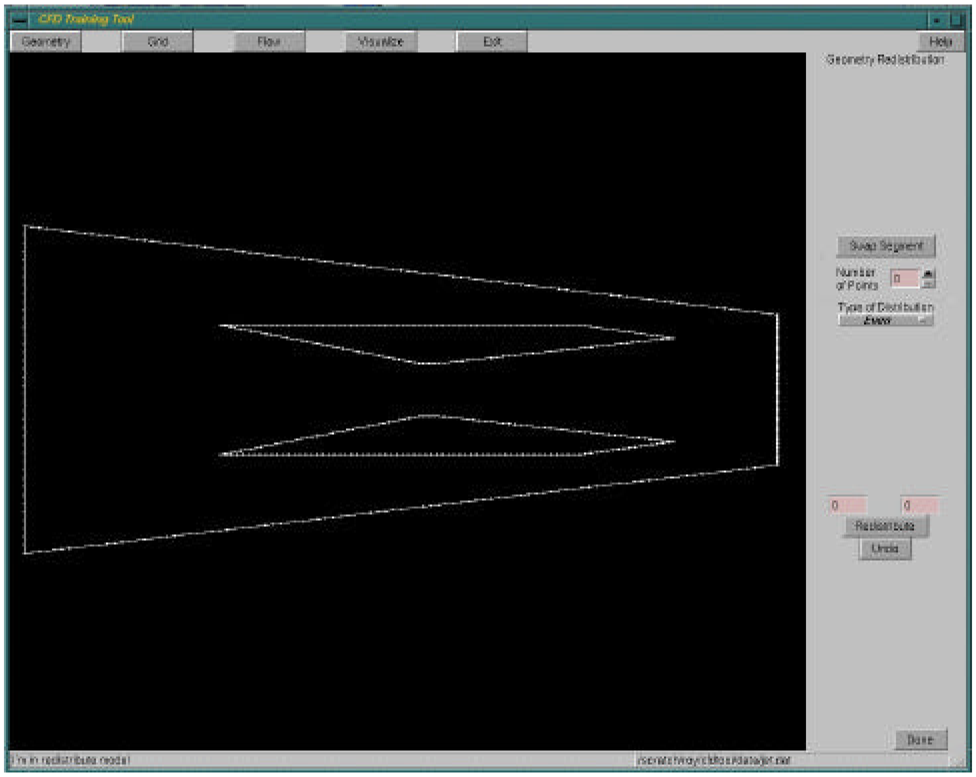


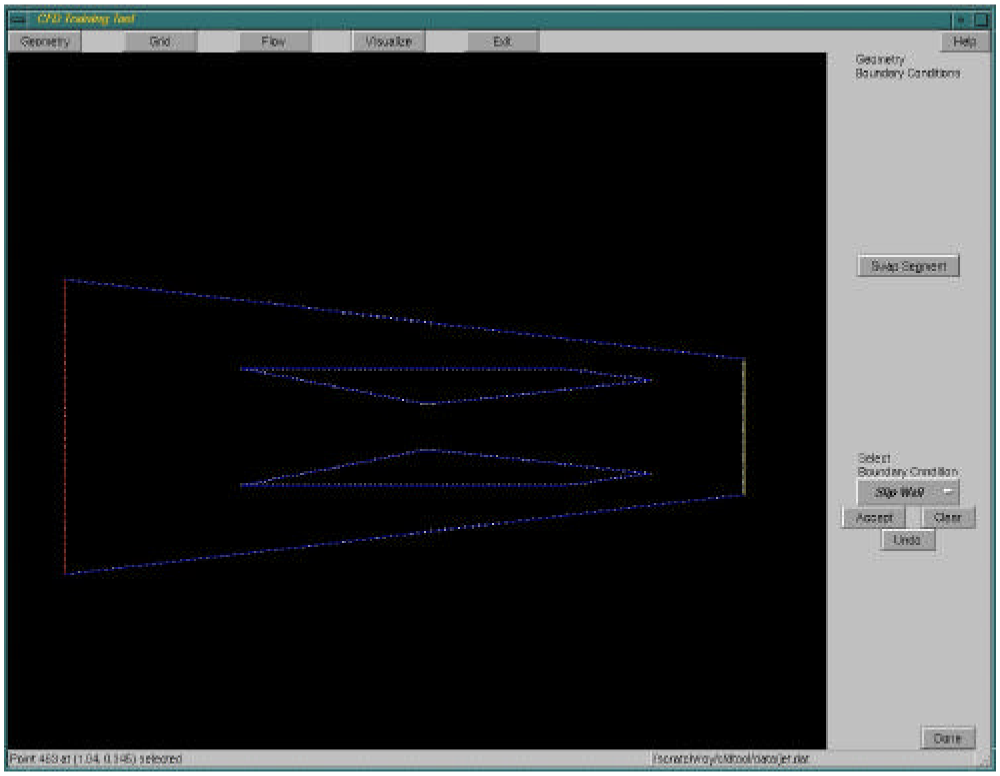
Figure 3. Geometry Module



I'm in redistribute mode!

/scratch/roy/cfdtool/data/jet.dat

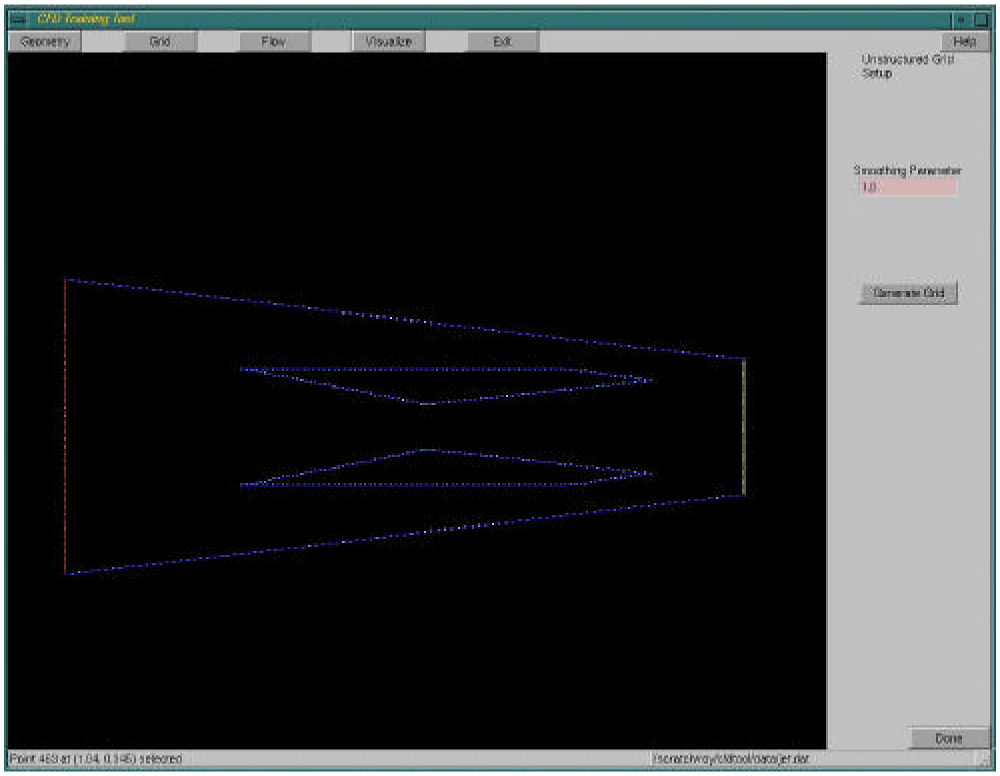
Figure 4. Module for Redistributing Points



Point 463 at (1.04, 0.345) selected

/scratch/roy/cfdtool/data/jet.dat

Figure 5: Module to Setup Boundary Condition



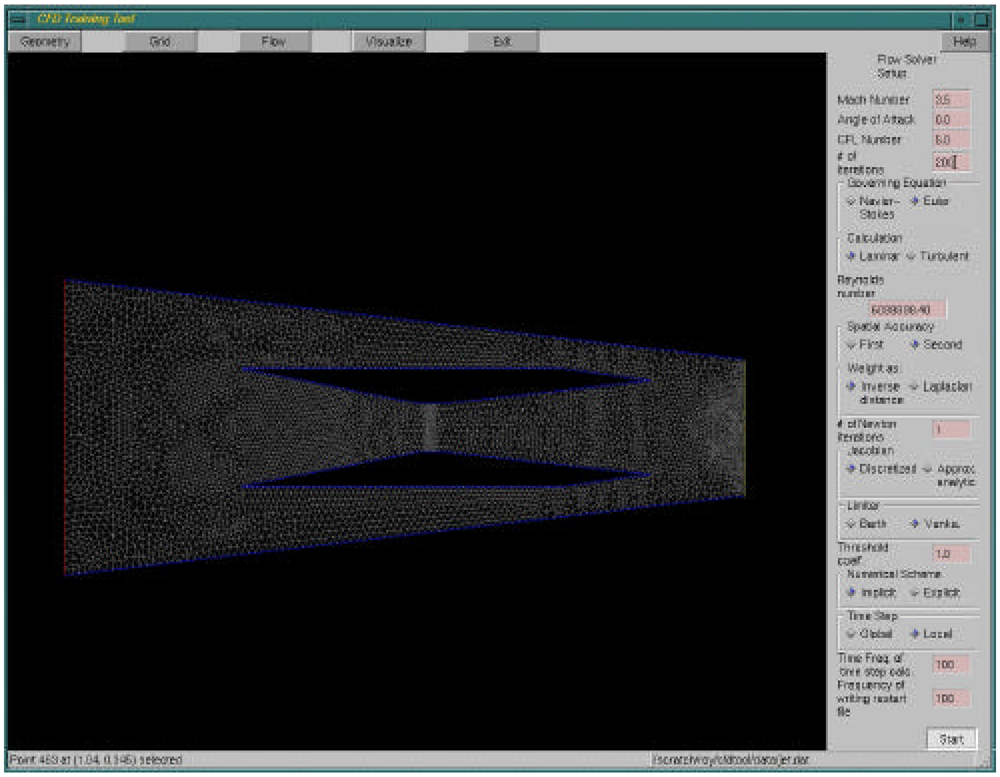
Unstructured Grid Setup

Smoothing Parameter: 1.0

Point 463 at (1.04, 0.145) selected

/scratch/wy/cfdtool/data/jet.dat

Figure 6. Unstructured Grid Generation Module



Flow Solver Setup

Mach Number: 3.5
Angle of Attack: 0.0
CFL Number: 6.0
# of iterations: 200

Governing Equation: Navier-Stokes [✓ Euler]

Calculation: [✓ Laminar] Turbulent

Reynolds number: 6088898.40

Spatial Accuracy: First [✓ Second]

Weight as: [✓ Inverse distance] Laplacian

# of Newton iterations: 1

Jacobian: [✓ Discretized] Approx. analytic

Limiter: Barth [✓ Venka.]

Threshold coeff: 1.0

Numerical Scheme: [✓ Implicit] Explicit

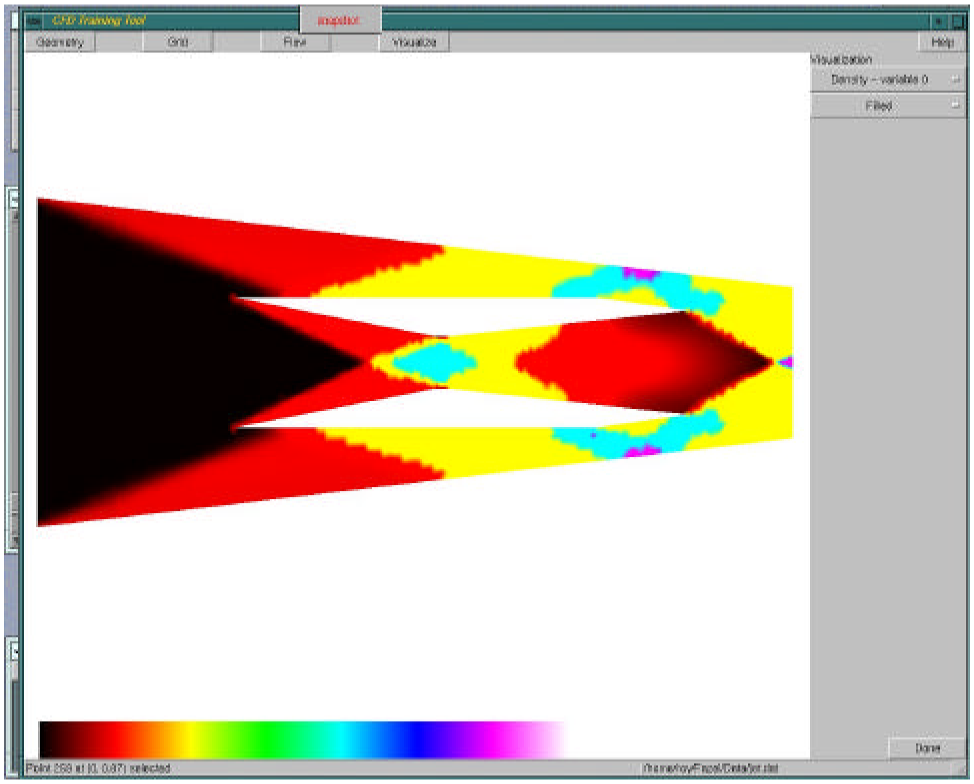
Time Step: Global [✓ Local]

Time Freq. of time step calc.: 100
Frequency of writing restart file: 100

Point 463 at (1.04, 0.145) selected

/scratch/wy/cfdtool/data/jet.dat

Figure 7: Flow Condition Setup Module



Point 259 at (0, 0.87) selected

/home/roy/Final/Data/jet.dat

Figure 8. Visualization Module

3 Technical Description of the Hybrid Grid Generator

The first type of generalized grid considered here is the so-called hybrid grid. The hybrid grid generation process is a combination of structured and unstructured grid methodologies. The hybrid grids used here consist of a structured grid in the viscous dominated regions and an unstructured grid in the rest of the field. Good quality structured grids in the boundary layer are generated by an advancing layer algorithm coupled with local elliptic smoothing (Huang [1]). The overlapped structured grid from the different solid bodies in the domain is trimmed by comparing the aspect ratios of the cells. The remainder of the domain is filled with an unstructured grid.

<u>3.a Generation of the Structured Grids</u>

An advancing front scheme is utilized to generate the structured grid. Marching fronts are started from the different entities in the field. The normal to the body surface at each point is calculated by averaging the normals of the line segments sharing the point. The point in the next layer is evaluated utilizing a user-specified distance along the normal. Once the second layer of points is fully determined, the second layer is taken as the new marching front. A third layer is generated from the previously generated grid line using the procedure outlined above. An elliptic solver is applied to these three grid lines to smooth grid lines and avoid grid crossing. This step is important for geometries having concave and convex surfaces. The smoothing will help ensure a smooth turning of the surface normals for concave surfaces. After the application of the elliptic solver, the second grid line is taken as the new front and the process is repeated until the required number of structured grid layers in the boundary layer is reached. Since this is a marching scheme and the application of the elliptic solver is local, the overall process is very fast and robust.

3.b Trimming of the Structured Grids

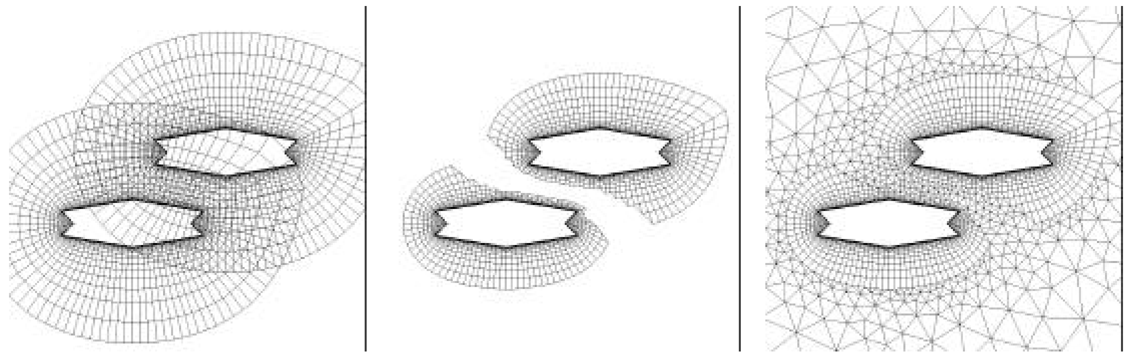
As discussed above, structured grids in the vicinity of the solid bodies in the domain are generated independently. These structured grids are trimmed based on the aspect ratio of the cells and overlapping of the component grids (Huang [1]). All cells that have an aspect ratio less than unity are removed. After clearing the cells with aspect ratio less than one and those that overlap, the points located near the cleared gap region are connected to form a closed loop. The closed outer boundary loops for the component grids together with the global outer boundary information are supplied as boundaries for the unstructured grid generation.

Sometimes two component grids may be close enough so that there is not enough space for growing the high aspect ratio cells to unity before the trimming. In this case, the boundary points are refined to get a smooth transition of the structured grid to unstructured grid.

3.c Generation of the Unstructured Grid

After trimming, part of the physical domain is filled with non-overlapping structured grids and the rest of the domain is left empty. This empty region is then filled with triangular cells. The unstructured grid is generated using Delaunay triangulation (Weatherill [2]). In Delaunay triangulation the boundaries are formed by the outer boundaries of trimmed structured grids and the outer boundary of the physical domain.
As the final step of the hybrid grid generation, the trimmed structured grid and the unstructured grid are connected to form a single grid. The connectivity table is also updated with the new node numbers.

An example of a hybrid grid generated using this approach is shown in Figure (9). Figure (10) shows a close-up view of the trimmed structured grid near the leading edge of an iced airfoil and the resulting hybrid grid is shown in Figure (11).



(a) Structured Grid Generation

(b) Trimming Structured Grid

(c) Filling the Domain with Unstructured Grid

Figure 9 Stages of Hybrid Grid Generation

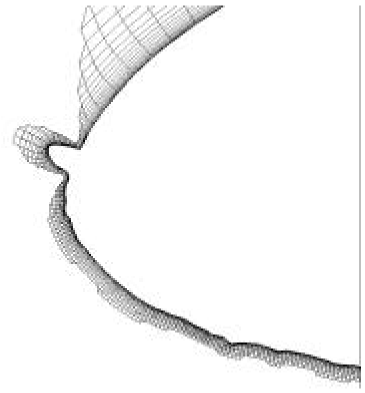


Figure 10 Trimmed Structured Grid Near the Leading Edge

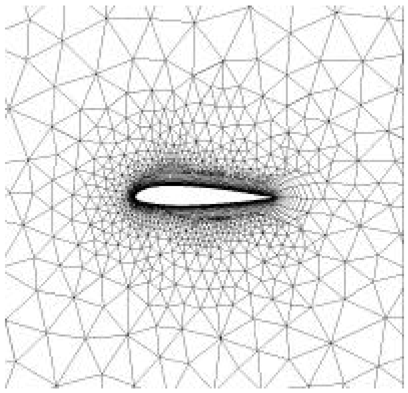


Figure 11 Hybrid Grid Around an Iced Airfoil

4 Technical Description of the Flow Solver

For the flow solver, the integral form of the non-dimensionalized Navier-Stokes equations is taken as the governing equation (Koomullil [3]) and is given below. The non-dimensionalizations are based on the freestream conditions. The velocity components are non-dimensionalized with respect to (w.r.t.) the total freestream velocity

$$\int_{\Omega} \frac{\partial Q}{\partial t}\, d\Omega + \oint_{\partial\Omega} F(Q) \cdot \vec{n}\, ds = \int_{\partial\Omega} F^{v}(Q) \cdot \vec{n}\, ds \qquad (1)$$

where Q is the conserved variable vector, $F(Q)$ is the inviscid flux vector, $F^v(Q)$ is the viscous flux vector, and n is the outward pointing unit normal to the control volume.

Finite-volume schemes are well suited for generalized grids, because a typical generalized grid is an agglomeration of polygons with different numbers of sides. For the present work, we have used a cell-centered, finite volume scheme, in which cell-averaged flow variables are stored at the cell center. The discretized form of the above equation can be written as:

$$\frac{\Delta Q}{\Delta t} V_i = -\sum_{j=1}^{k} F_{ij} \cdot \vec{n} ds_j + \sum_{j=1}^{k} F_{ij}^{v}(Q) \cdot \vec{n} ds_j \qquad (2)$$

where Vi is the volume of the cell, i, j represents the edges that form the cell, and F_{ij} and $F^v{}_{ij}$ are the inviscid and viscous numerical fluxes, respectively.

The inviscid numerical flux passing through the cell faces is calculated by Roe's approximate Riemann solver (Roe [4]) as an exact solution for a linearized Riemann problem. Using the approximate Riemann solver, the flux through a cell face is given by

$$F_{ij} = \frac{1}{2}\left[F(Q_L) + F(Q_R) - \left|\bar{A}\right|\left(Q_R - Q_L\right)\right] \qquad (3)$$

where subscripts L and R represent cells on the left and right sides of the cell-face, and $|\bar{A}|$ is the Roe-averaged matrix (Roe [4]).

Higher-order accuracy in the spatial discretization is obtained using a linear reconstruction of the conserved variables, which are themselves obtained using a Taylor series expansion and Gauss' theorem.

The Taylor's series expansion for a function of two variables is written as

$$Q(x,y) = Q(x_i, y_i) + (\nabla Q)_{(x_i,y_i)} \cdot \Delta\vec{r} + O(\Delta\vec{r}^{\,2}) \tag{4}$$

The gradient of the conserved variables at the cell center is estimated using Green's theorem with the control volume taken as the cell itself. The values of the conserved variable vector at the nodes are estimated based on a weighted averaging procedure. During the reconstruction process, local extrema may be created. These extrema may produce spurious values in regions where there are sharp jumps in the flow variables such as shocks, contact discontinuities, expansion regions, etc. In order to avoid this, a limiter function is employed and the Taylor's series expansion is modified as

$$Q(x,y) = Q(x_i, y_i) + f(\nabla Q)_{(x_i,y_i)} \cdot \Delta\vec{r} + O(\Delta\vec{r}^{\,2}) \tag{5}$$

where f is the limiter function (Barth [5], Venkatakrishnan [6]) and its value is limited between zero and one.

In the case of implicit schemes, the numerical flux crossing the cell face is a function of the conserved variables at the *(n+1)*th time level. The flux vector has to be linearized before the evaluation of the flux crossing the cell faces. The resulting linear system can be written as,

$$\left[\frac{V_i}{\Delta t} I + \sum_{j=1}^{k}\left(\frac{\partial H_{ij}}{\partial Q_i}\right)^n\right]\Delta Q_i^n + \sum_{j=1}^{k}\left[\left(\frac{\partial H_{ij}}{\partial Q_j}\right)^n \Delta Q_j^n\right] = -\Re_i^n \tag{6}$$

where $H_{ij} = F_{ij} \cdot \vec{n}ds$

The Jacobian matrices $\dfrac{\partial H_{ij}}{\partial Q_i}$ and $\dfrac{\partial H_{ij}}{\partial Q_j}$ can be estimated using approximate analytical Jacobians, by taking the Roe-averaged matrix $|\bar{A}|$ to be constant, or by a numerical approach (Whitfield [7]). During the linearization, the viscous flux vectors are treated explicitly. The matrix system resulting from the above equation is solved using the Generalized Minimum RESidual (GMRES) method (Saad and Schultz [8]).

The simulation of many complex features of flows of practical importance needs to account for the flow's turbulent behavior. The laminar viscosity is usually a function of temperature and is estimated using Sutherland's formula (Warsi [9]). The turbulent viscosity is a function of the flow and is usually evaluated using an empirical model. In the present study, the turbulent viscosity is estimated using the Spalart-Allmaras one-equation turbulence model (Spalart and Allmaras [10]) and the Reynolds stress is modeled using the Boussinesq hypothesis (Warsi [9]). The Spalart-Allmaras one-equation model encompasses a solution of a second-order partial differential equation for the variable $\bar{u}$. The turbulent kinematic viscosity is estimated by applying a damping function.

The non-dimensional form of the Spalart-Allmaras one equation turbulence model in the vector invariant form, without tripping terms, can be written as

$$\frac{\partial \bar{u}}{\partial t} = \vec{V} \cdot \nabla \bar{u} + C_{b1} \tilde{S} \bar{u} + \frac{1}{s\,\mathrm{Re}_L} \left(\nabla \cdot \left(u + \bar{u} \right) \nabla \bar{u} \right) + \frac{C_{b2}}{s\,\mathrm{Re}_L} \left(\nabla \bar{u} \right)^2 - \frac{C_{v1} f_v}{\mathrm{Re}_L} \left(\frac{\bar{u}}{d} \right)^2 \qquad (7)$$

The different variables and functions appearing in the above equation can be summarized as

$$u_t = \bar{u} f_{v1} \qquad f_{v1} = \frac{c^3}{c^3 + C_{v1}^3} \qquad c = \frac{\bar{u}}{u}$$

$$f_{v2} = 1 - \frac{c}{1 + c f_{v1}} \qquad \tilde{S} = S + \frac{1}{\mathrm{Re}_L} \left(\frac{\bar{u}}{k^2 d^2} \right) f_{v2}$$

$$f_v = g \left[\frac{1 + C_{v3}^6}{g^6 + C_{v3}^6} \right]^{\frac{1}{6}} \qquad g = r + C_{v2} \left(r^6 - r \right)$$

$$r = \frac{1}{\mathrm{Re}_L} \left(\frac{\bar{u}}{\tilde{S} k^2 d^2} \right) \qquad C_{v1} = \frac{C_{b1}}{k^2} + \frac{1 + C_{b2}}{s}$$

$C_{b1}=0.1355$ $\quad s=2/3$ $\quad C_{b2}=0.622$ $\quad k=0.41$

$C_{w2}=0.3$ $\quad C_{w3}=2.0$ $\quad C_{v1}=7.1$

The details of the implementation of Spalart-Allmaras one equation model can be found in Koomullil [3].

5 Future Work

At present, all modules described in the Section 2 are loosely incorporated and linked with appropriate programs. Streamlining of the process is under development. Current work in progress includes: creating a unified geometry definition for unstructured and hybrid grids, improving revision capabilities from the visualization module to geometry modules for iteration purposes, improving memory management, and producing better color maps for visualization purposes. Also, in the current version of CFDTool, grid generation is limited to only unstructured grids. Work is in progress to link the hybrid grid generator described in Section 3 of this report into the CFDTool.

6 References

[1] Huang, Chih-Ti., "Hybrid Grid Generation System," Master's Thesis, Department of Aerospace Engineering, Mississippi State University, MS, August 1996.

[2] Weatherill, N. P., "Grid Generation by Delaunay Triangulation," Lecture Series in von Karman Institute for Fluid Dynamics 1993-94.

[3] Koomullil, R. P., "Flow Simulation System for Generalized Static and Dynamic Grids," Ph.D. Dissertation, Department of Aerospace Engineering, Mississippi State University, MS, May 1997.

[4] Roe, P. L., "Approximate Riemann Solvers, Parameter Vector, and Difference Schemes," Journal of Computational Physics, Vol. 43, 1981, pp. 357-372.

[5] Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured meshes," AIAA Paper 89-0366, Jan. 1989.

[6] Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady State Solutions," AIAA Paper 93-0880, Jan. 1993.

[7] Whitfield, D. L., and Taylor, L., "Discretized Newton-Relaxation Solution of High Resolution Flux-Difference Split Schemes," AIAA Paper 91-1539, June 1991.

[8] Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Non-symmetric Linear Systems," SIAM Journal of Sci. Stat. Comp. Vol. 7, 1986, pp. 865-869.

[9] Warsi, Z. U. A., "Fluid Dynamics: Theoretical and Computational Approaches," CRC Press, 1993.

[10] Spalart. P. R., and Allmaras, S. R., "A One Equation Turbulence Model for Aerodynamic Flows," AIAA Paper 92-0439, Jan. 1992.

Acknowledgements

Authors would like to express their sincere appreciations to Mr. Kenneth Fazel of Mississippi State University, for implementing GUIs for CFDTool and to Dr. Nathan Prewitt, CFD onsite lead at ERDC, for his valuable suggestions and demonstration of the CFDTool to the ERDC community.